Date
4/9/21

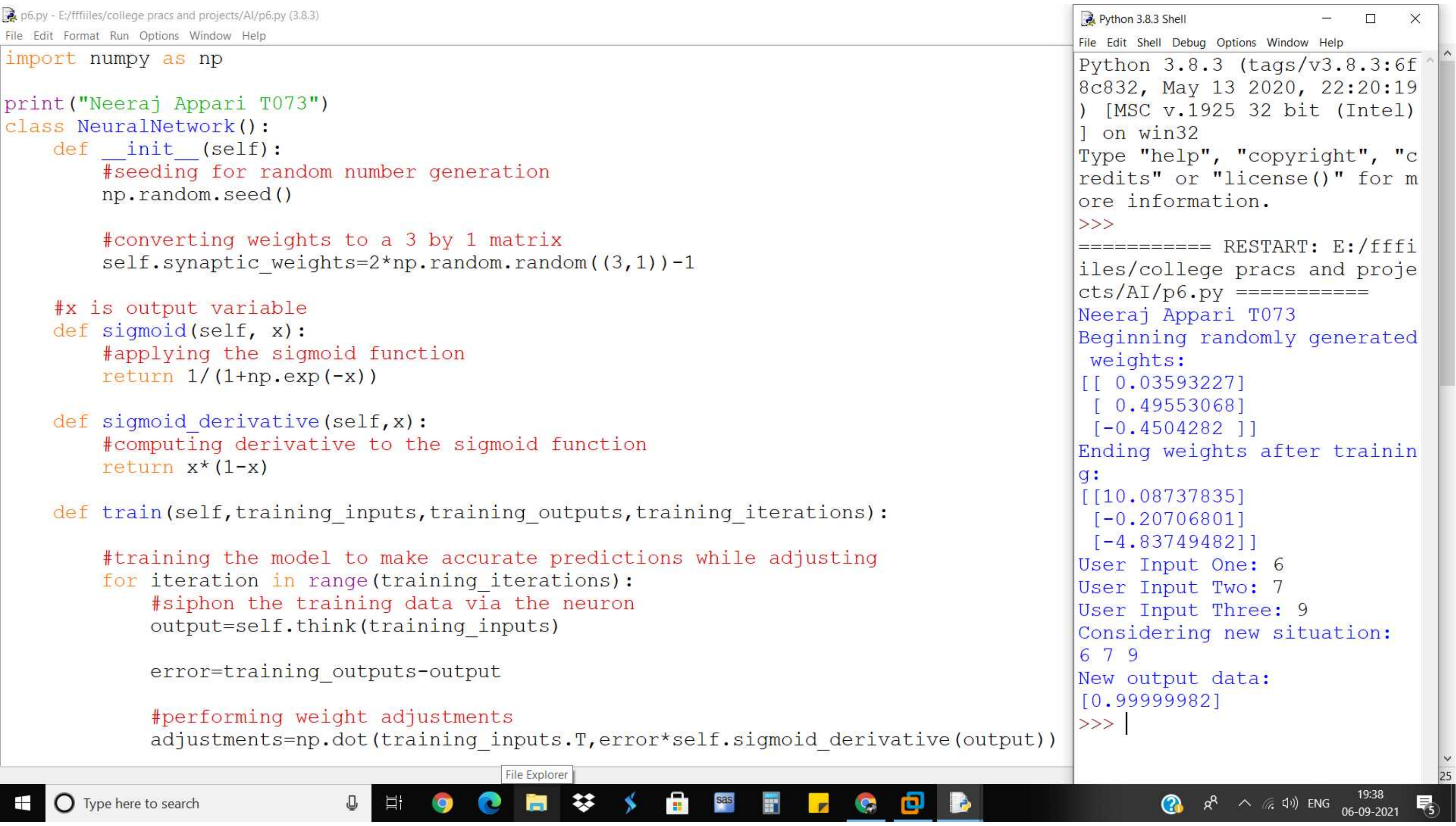
Name-Neeraj Arora ~~Arora~~ 7023

SDS Page No.
Date / /

AI Practical 6

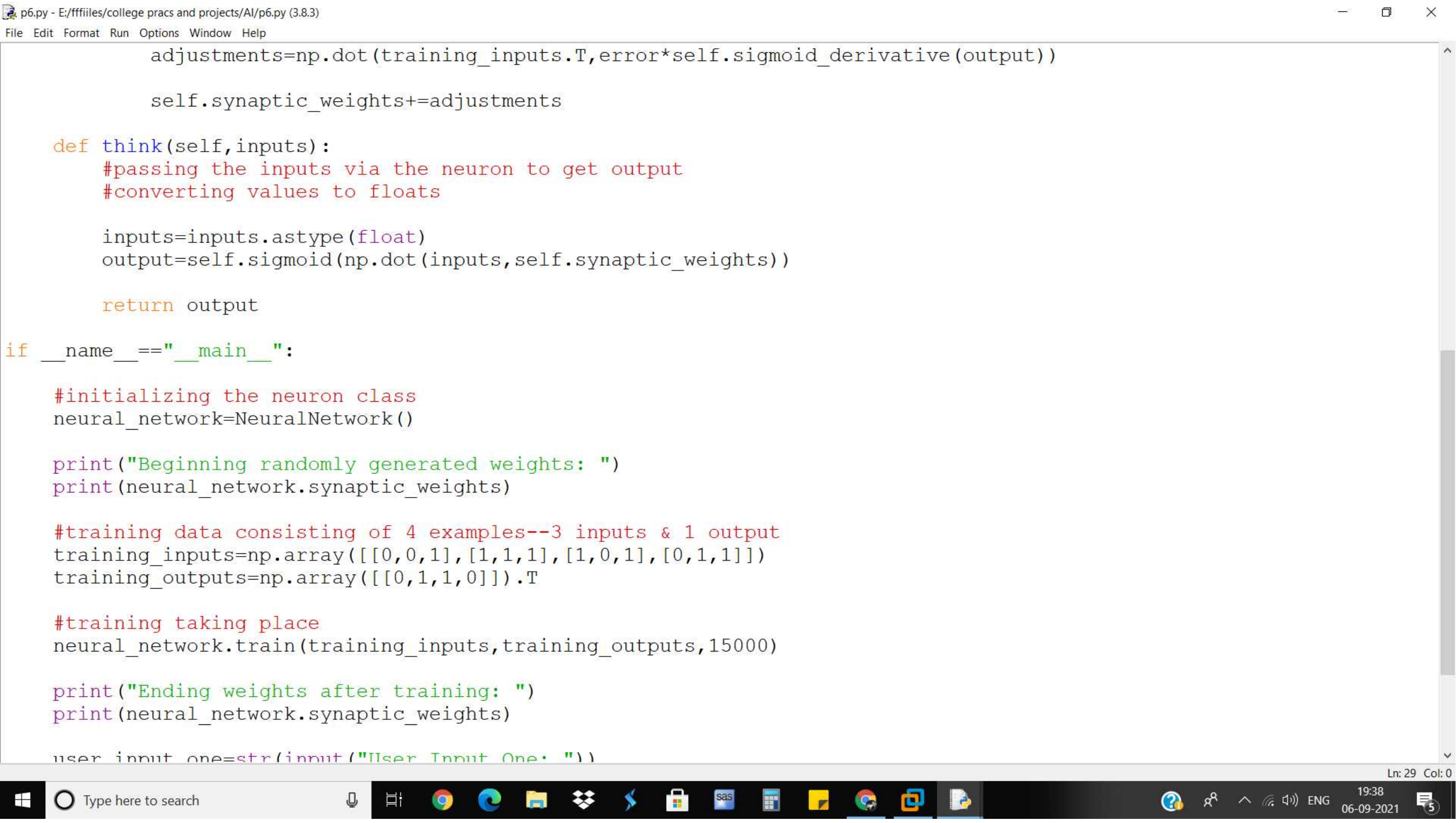
Aim - Implement feed-forward back propogation neural network learning algorithm

1) Seeding to ~~ram~~random number generator
2) Converting weights to 3 by matrix
3) x is output variable
4) Applying sigmoid function
5) Computing derivative to the sigmoid function.
6) Turning the model to make accurate predication while adjusting
7) Siphon the training data via the neuron
8) Passing the inputs via the neuron to get output & converting values to floats
9) Performing weight adjustment
10) Initialize the neuron class
11) Training data consisting of example & 3 input & output and training taking place

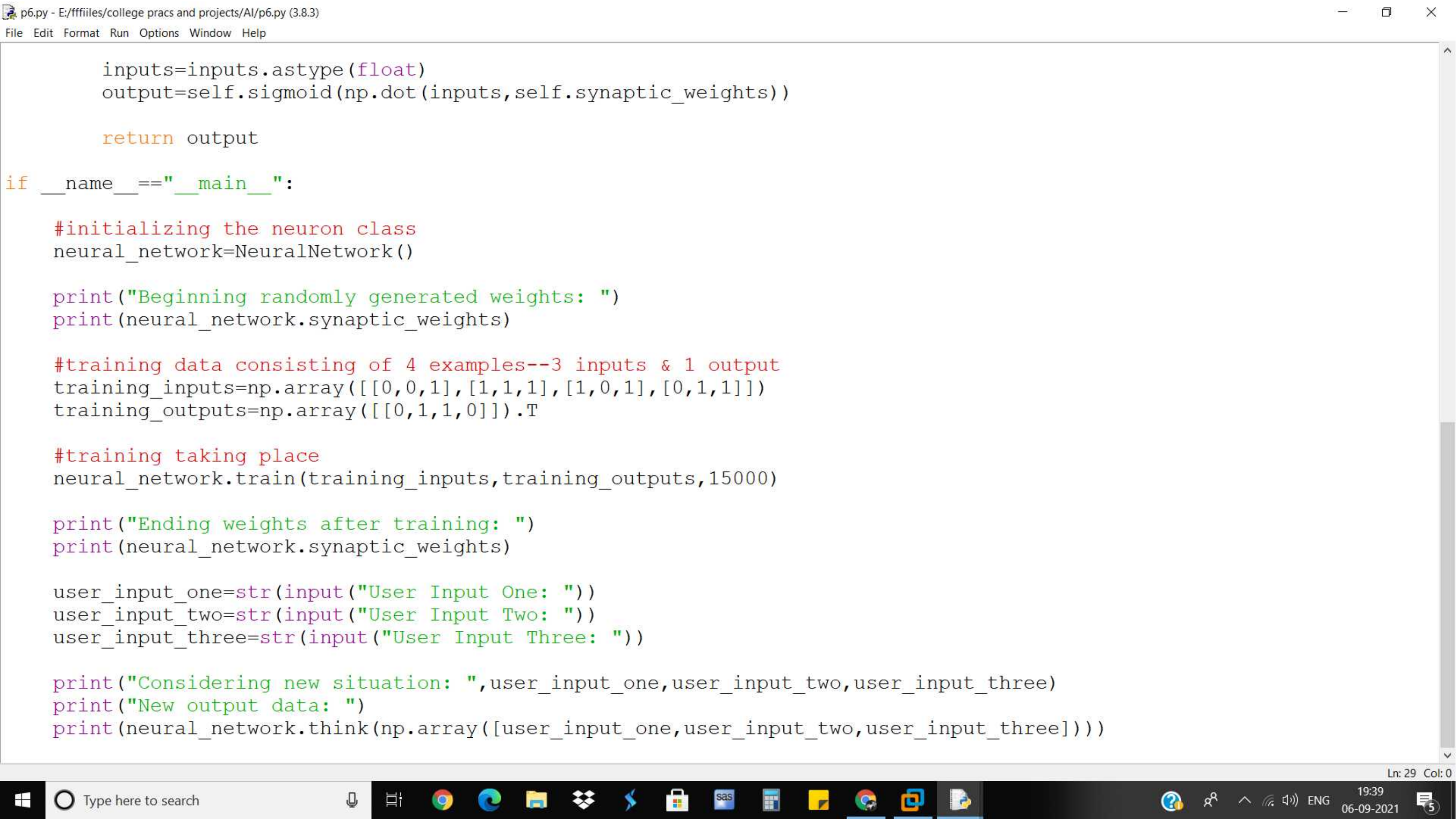

p6.py - E:/fffiiles/college pracs and projects/AI/p6.py (3.8.3)
File Edit Format Run Options Window Help
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
File Explorer
Type here to search
ENG
19:38
06-09-2021

```
import numpy as np

print("Neeraj Appari T073")
class NeuralNetwork():
    def __init__(self):
        #seeding for random number generation
        np.random.seed()

        #converting weights to a 3 by 1 matrix
        self.synaptic_weights=2*np.random.random((3,1))-1

    #x is output variable
    def sigmoid(self, x):
        #applying the sigmoid function
        return 1/(1+np.exp(-x))

    def sigmoid_derivative(self,x):
        #computing derivative to the sigmoid function
        return x*(1-x)

    def train(self,training_inputs,training_outputs,training_iterations):

        #training the model to make accurate predictions while adjusting
        for iteration in range(training_iterations):
            #siphon the training data via the neuron
            output=self.think(training_inputs)

            error=training_outputs-output

            #performing weight adjustments
            adjustments=np.dot(training_inputs.T,error*self.sigmoid_derivative(output))
```

```
Python 3.8.3 (tags/v3.8.3:6f
8c832, May 13 2020, 22:20:19
) [MSC v.1925 32 bit (Intel)
] on win32
Type "help", "copyright", "c
redits" or "license()" for m
ore information.
>>>
=========== RESTART: E:/fffi
iles/college pracs and proje
cts/AI/p6.py ===========
Neeraj Appari T073
Beginning randomly generated
 weights:
[[ 0.03593227]
 [ 0.49553068]
 [-0.4504282 ]]
Ending weights after trainin
g:
[[10.08737835]
 [-0.20706801]
 [-4.83749482]]
User Input One: 6
User Input Two: 7
User Input Three: 9
Considering new situation:
6 7 9
New output data:
[0.99999982]
>>> 
```


p6.py - E:/fffiiles/college pracs and projects/AI/p6.py (3.8.3)
File Edit Format Run Options Window Help
Ln: 29 Col: 0
Type here to search
ENG
19:38
06-09-2021

```
            adjustments=np.dot(training_inputs.T,error*self.sigmoid_derivative(output))

            self.synaptic_weights+=adjustments

    def think(self,inputs):
        #passing the inputs via the neuron to get output
        #converting values to floats

        inputs=inputs.astype(float)
        output=self.sigmoid(np.dot(inputs,self.synaptic_weights))

        return output

if __name__=="__main__":

    #initializing the neuron class
    neural_network=NeuralNetwork()

    print("Beginning randomly generated weights: ")
    print(neural_network.synaptic_weights)

    #training data consisting of 4 examples--3 inputs & 1 output
    training_inputs=np.array([[0,0,1],[1,1,1],[1,0,1],[0,1,1]])
    training_outputs=np.array([[0,1,1,0]]).T

    #training taking place
    neural_network.train(training_inputs,training_outputs,15000)

    print("Ending weights after training: ")
    print(neural_network.synaptic_weights)

    user_input_one=str(input("User Input One: "))
```



p6.py - E:/fffiiles/college pracs and projects/AI/p6.py (3.8.3)
File Edit Format Run Options Window Help
Ln: 29 Col: 0
Type here to search
ENG
19:39
06-09-2021

```
        inputs=inputs.astype(float)
        output=self.sigmoid(np.dot(inputs,self.synaptic_weights))

        return output

if __name__=="__main__":

    #initializing the neuron class
    neural_network=NeuralNetwork()

    print("Beginning randomly generated weights: ")
    print(neural_network.synaptic_weights)

    #training data consisting of 4 examples--3 inputs & 1 output
    training_inputs=np.array([[0,0,1],[1,1,1],[1,0,1],[0,1,1]])
    training_outputs=np.array([[0,1,1,0]]).T

    #training taking place
    neural_network.train(training_inputs,training_outputs,15000)

    print("Ending weights after training: ")
    print(neural_network.synaptic_weights)

    user_input_one=str(input("User Input One: "))
    user_input_two=str(input("User Input Two: "))
    user_input_three=str(input("User Input Three: "))

    print("Considering new situation: ",user_input_one,user_input_two,user_input_three)
    print("New output data: ")
    print(neural_network.think(np.array([user_input_one,user_input_two,user_input_three])))
```